New features of the batch system monitoring tool LLview

W.Frings

W.Frings@fz-juelich.de, llview.zam@fz-juelich.de

John von Neumann Institute for Computing
Central Institute for Applied Mathematics
Research Centre Jülich

ScicomP13, Garching, 20. July 2007

Overview

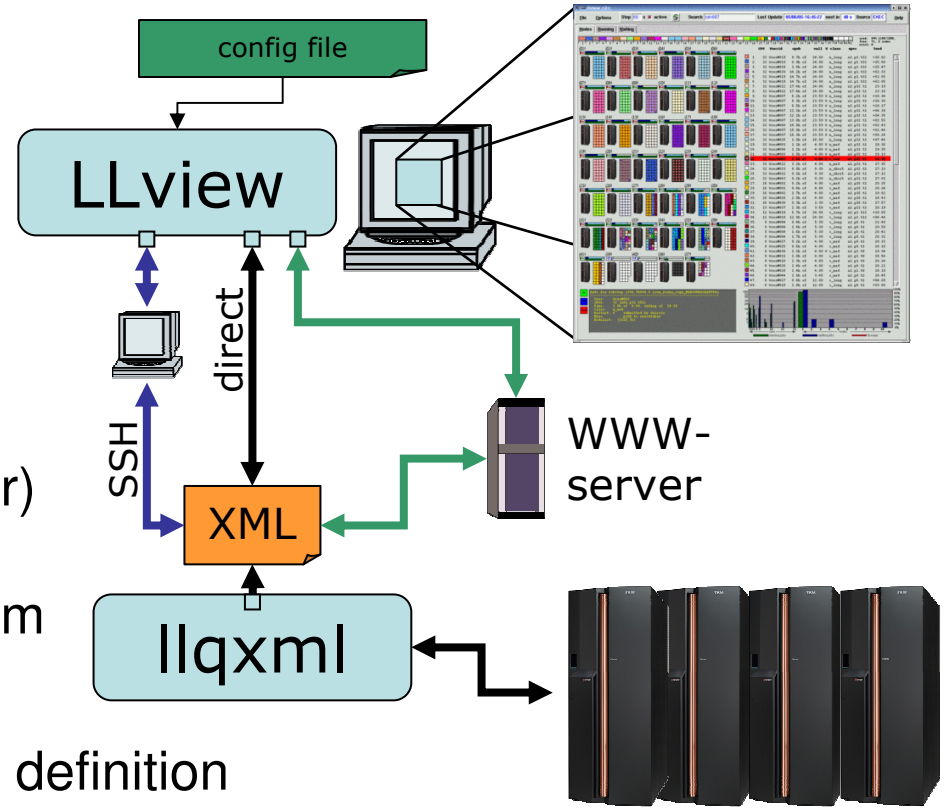
- Part I:
  - Introduction: Monitoring batch system usage
  - LLview components
  - Data access:
    - llqxml: Accessing batch system status with LoadLeveler API
    - Grid enabled data access (OGF RUS)
  - Multi-cluster display
  - Job scheduling prediction
  - Availability, installation and implementation of LLview
- Part II:
  - Online Demonstration
    - LLview displaying Jump and Jubl, Layout configuration
    - Prediction of system usage by scheduler simulation
    - Display of multi-cluster usage

Why monitoring batch systems?

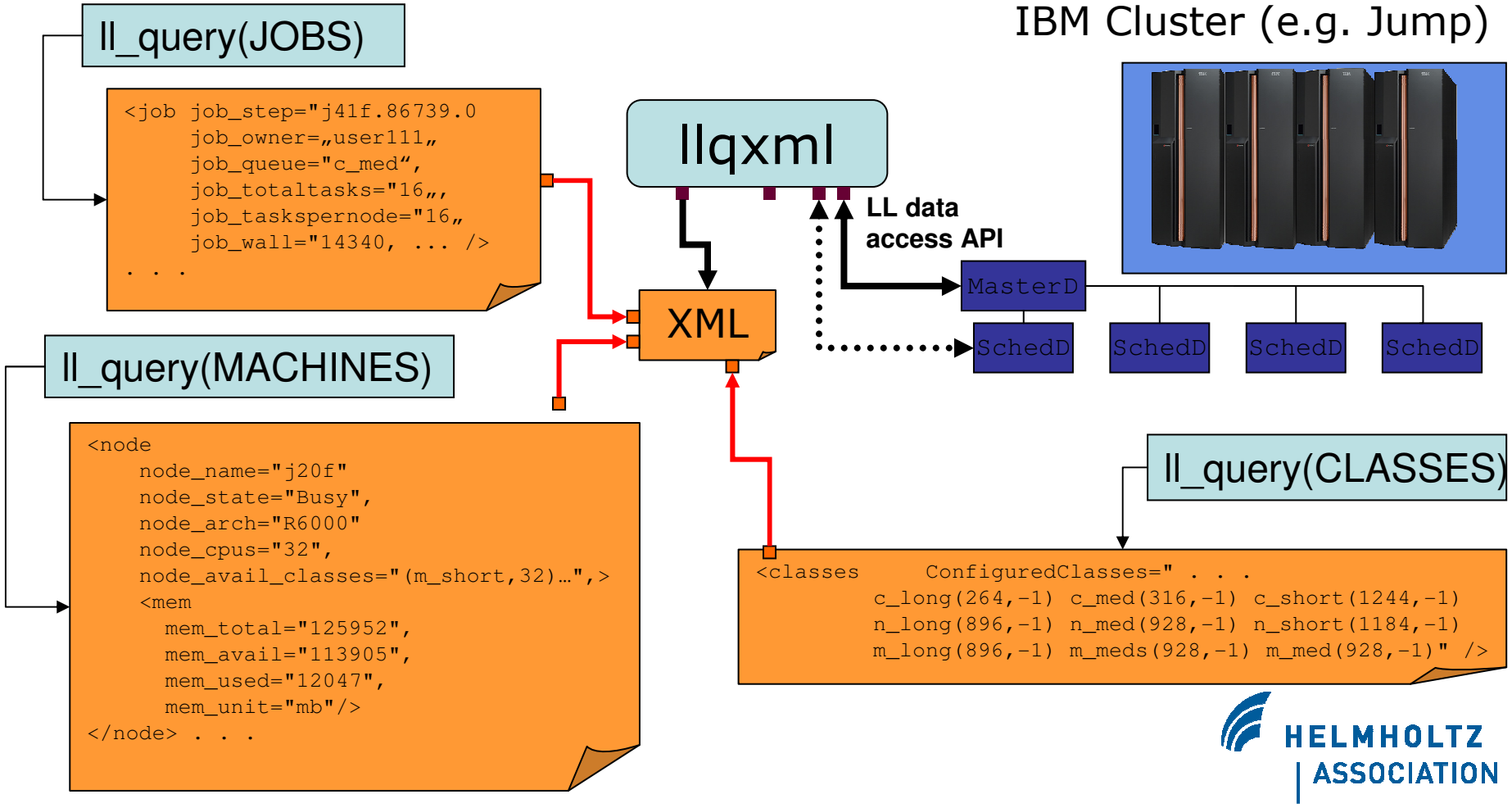
- For **administrators**:
  - Global overview of system usage
  - Throughput optimization
  - Batch system configuration optimization
  - Adaptive change of scheduling parameters
- For **users**:
  - Control over own running and waiting jobs
  - Planning of job submissions depending of current system usage, (job size, number of jobs)
- Why **LLview**?
  - Compact display of all usage data in one window
  - Easy access to a multitude of detailed batch system data via mouse sensitive information display

Components of LLview

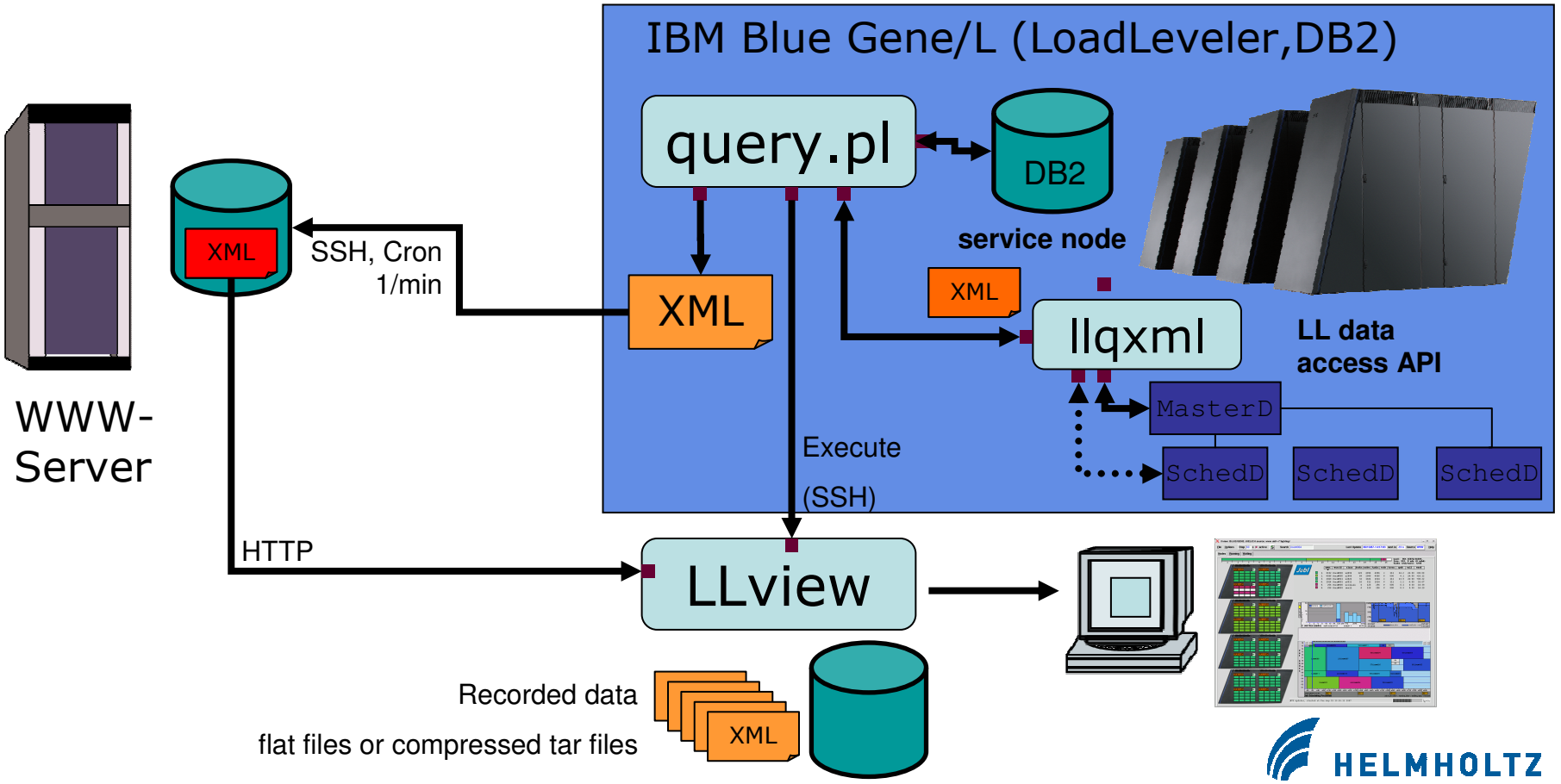
- Graphical client LLview
  - stand-alone application
  - on local workstation or front-end node
  - implemented in Perl/Tk
- Server llqxml
  - for data access (LoadLeveler)
  - runs on supercomputer
  - needs access to batch system
- Data format
  - XML, LLview document type definition
- Data access
  - HTTP, SSH, or direct execution on same system



Data Access LoadLeveler

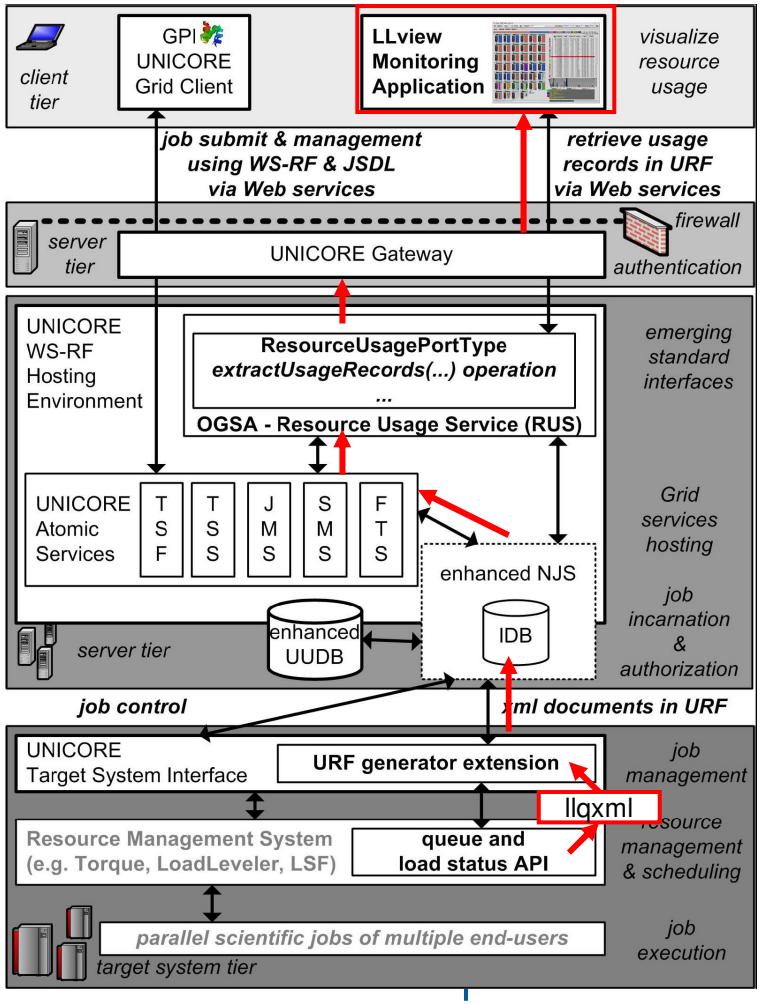


Data Flow, BlueGene/L



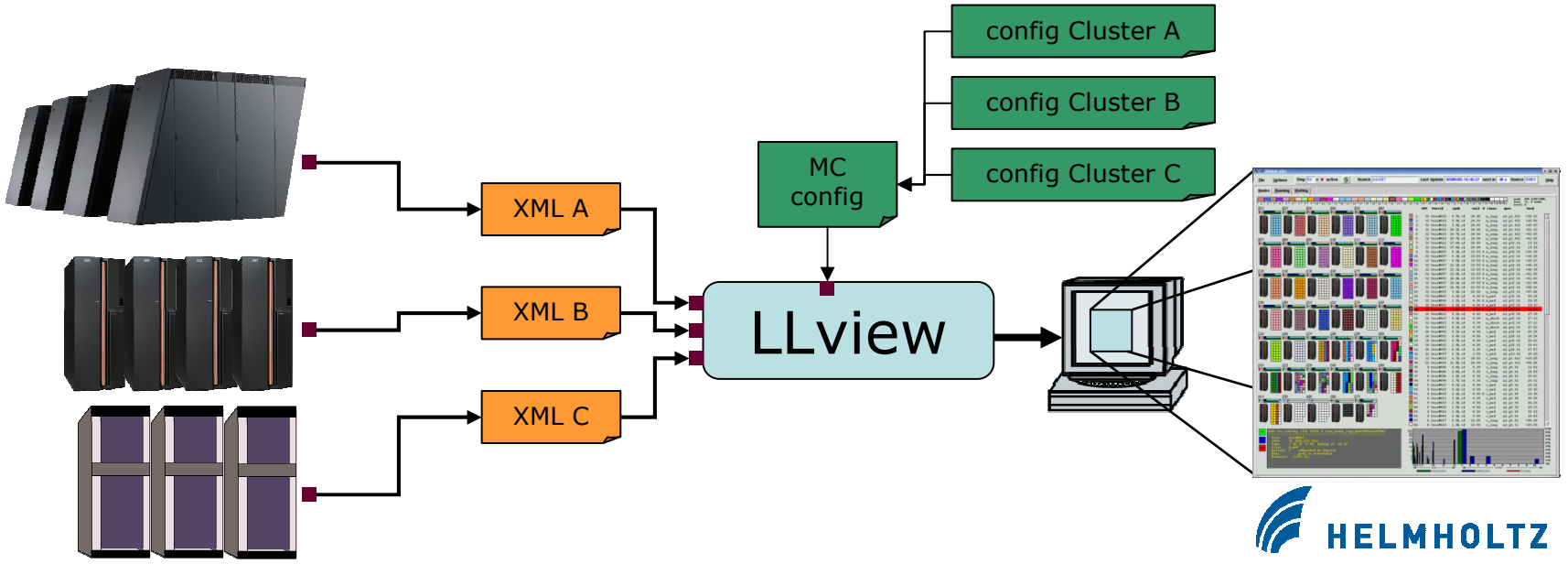
Grid enable data access

- Grid Middleware Unicore 6.0:
  - Seamless, secure and intuitive access to distributed grid resources
  - new Web-service based version
- Implementation in Unicore
  - in OMII Europe
  - OGF Resource Usage Service (RUS)
  - OGF Usage Record Format (URF) compliant XML documents
  - Integration of LLview data access in Unicore Target System Interface (TSI)
- Implementation in LLview
  - direct access to Web service using SOAP::Lite
  - converter for URF → LLview XML format



Multi-Cluster LLview

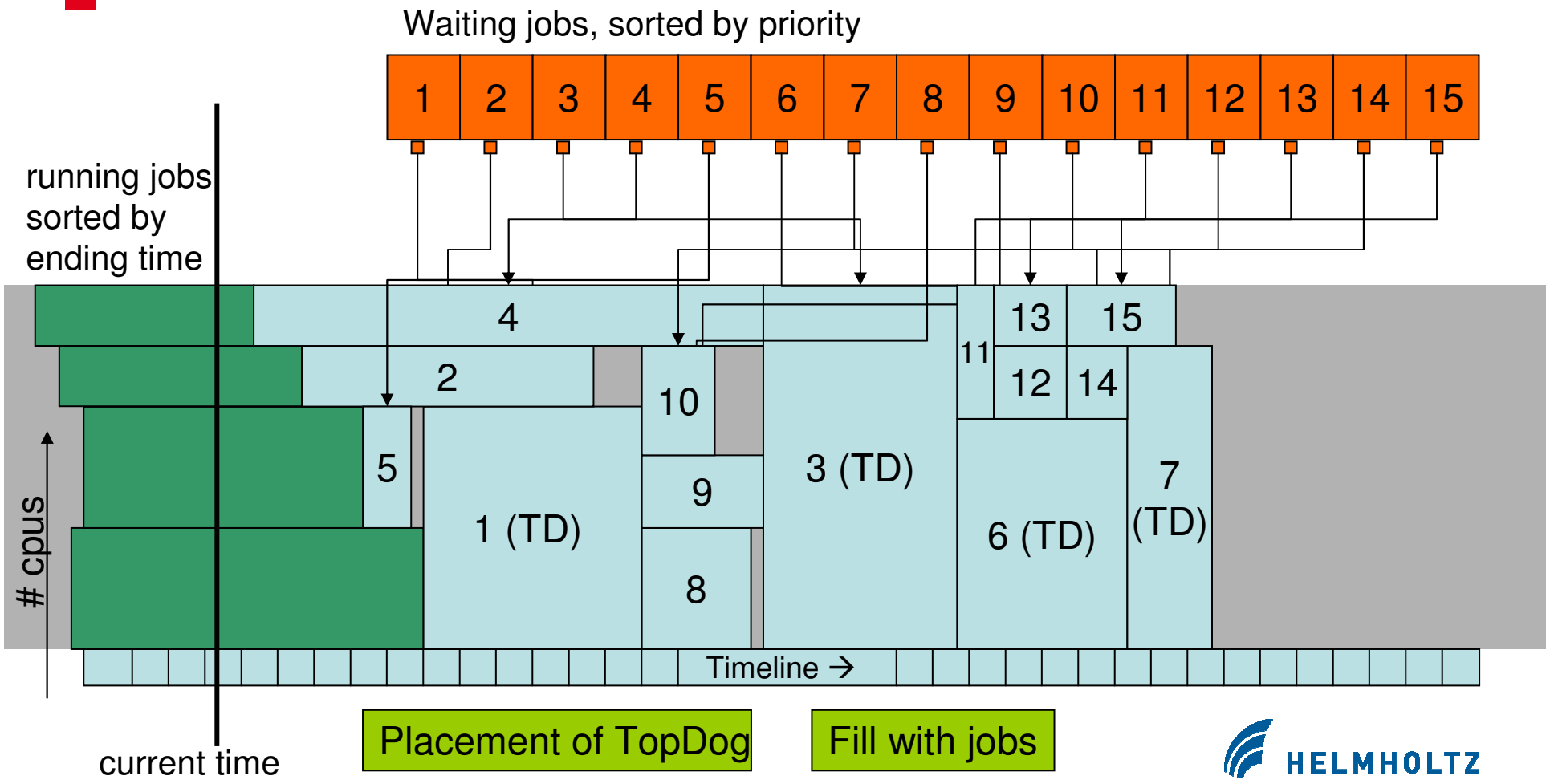
- Monitoring of more than one cluster in one window
  - Option `-mc` enables multi-cluster
  - one config file for each cluster,
    master-config file for the multi cluster, references to other config files
  - cluster specific data access methods possible



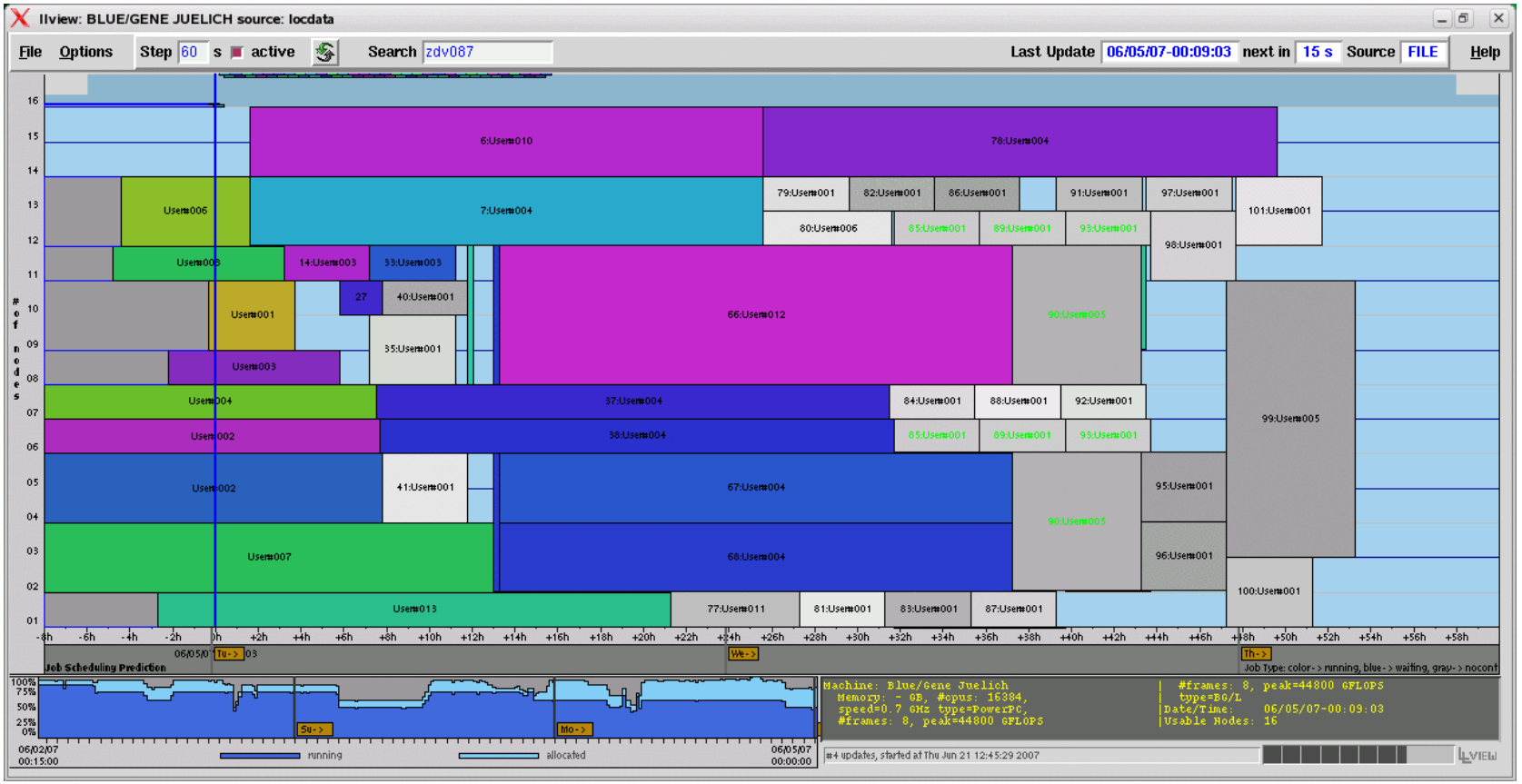
Prediction of Scheduling

- Scheduler simulation based on
  - current list of waiting jobs
  - job system priority given by LoadLeveler
  - maximum number of jobs per class/user
  - maximum starters (processes) on nodes
  - Backfilling rules, e.g. number of Topdogs
- Implementation
  - module in LLview, integrated in LLview client data processing
  - stand-alone utility for processing XML-files on server side
- Future Work
  - direct calculation of job priority based on user-specified formula and job specific limits and parameters
  - advanced reservations

Prediction of Scheduling (II)



Example: Predicition



number of nodes

now

timeline

Implementation & Availability

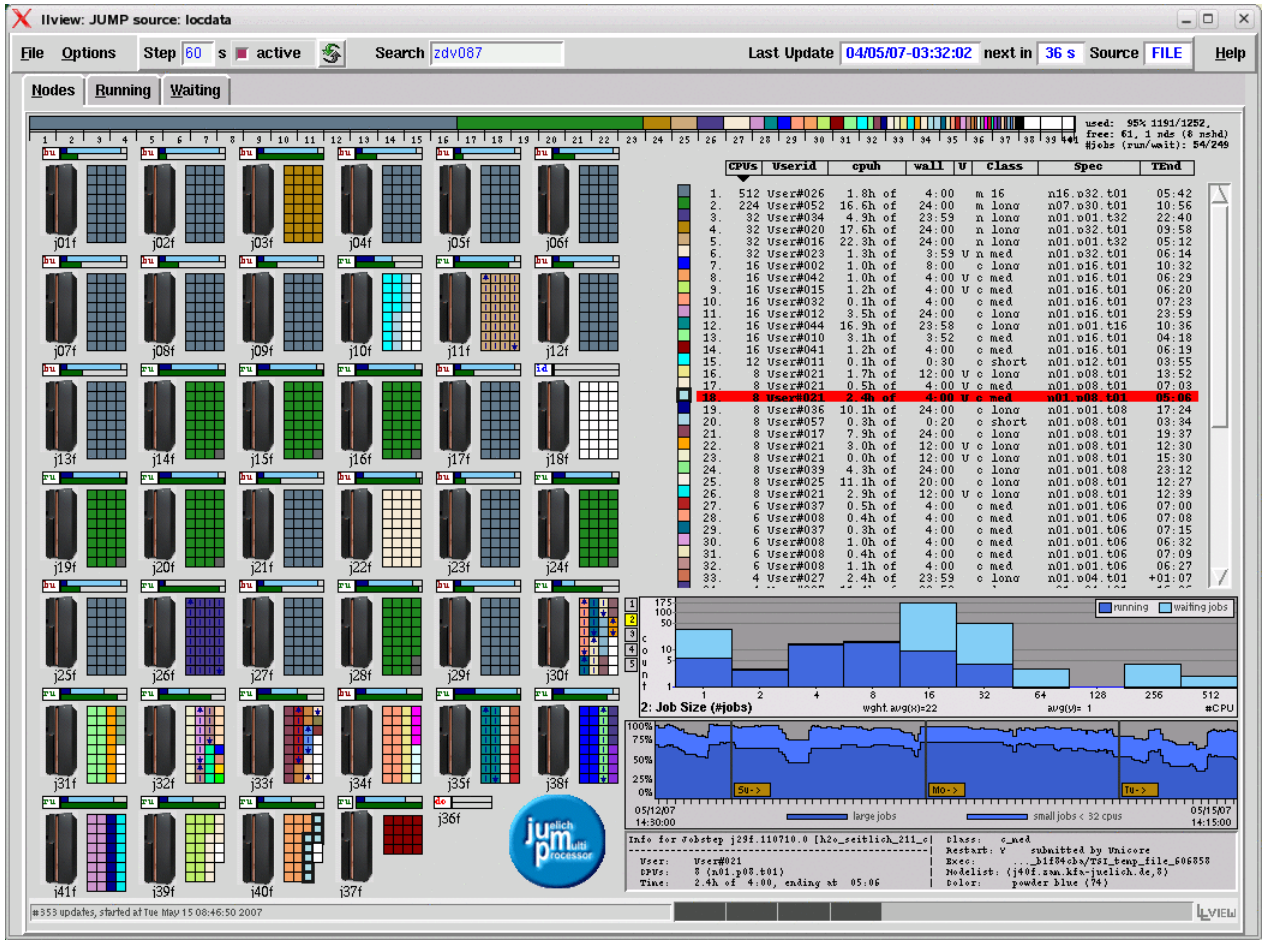
Implementation:

- Language: Perl/TK and C (25k lines of code)
- Client platforms:
  - Unix, Linux based (perl 5.6.x, perl 5.8.x, ...)
  - Windows XP, … (ActivePerl 5.8.x)
- Server modules for:
  - LoadLeveler (AIX/Linux),
  - BlueGene/L DB2,
  - PBSpro, Torque, and
  - LSF (TU Dresden)

Download:

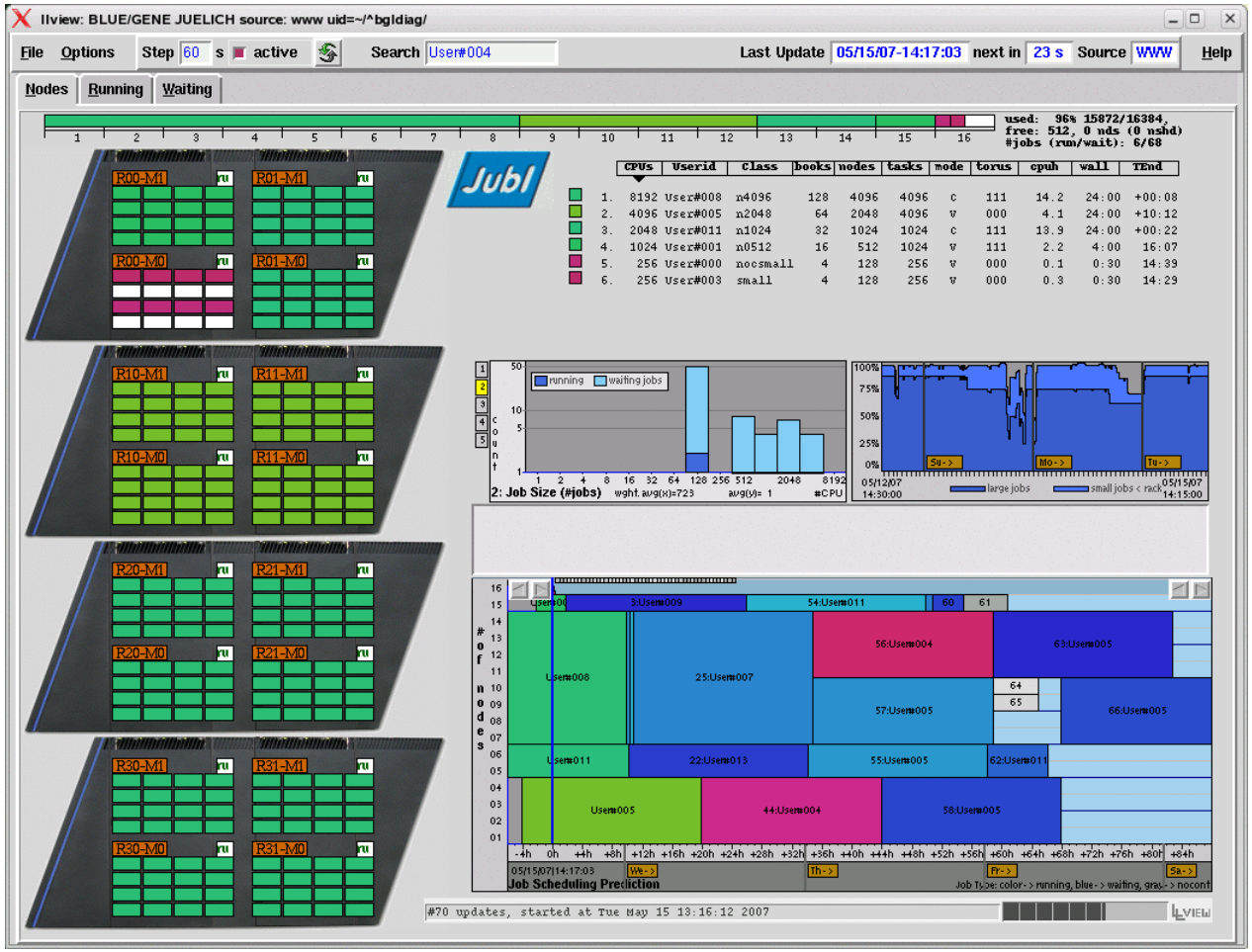
- free download: http://www.fz-juelich.de/zam/llview
- Public License, Registration
- current version: 1.2

Example 1: Jump



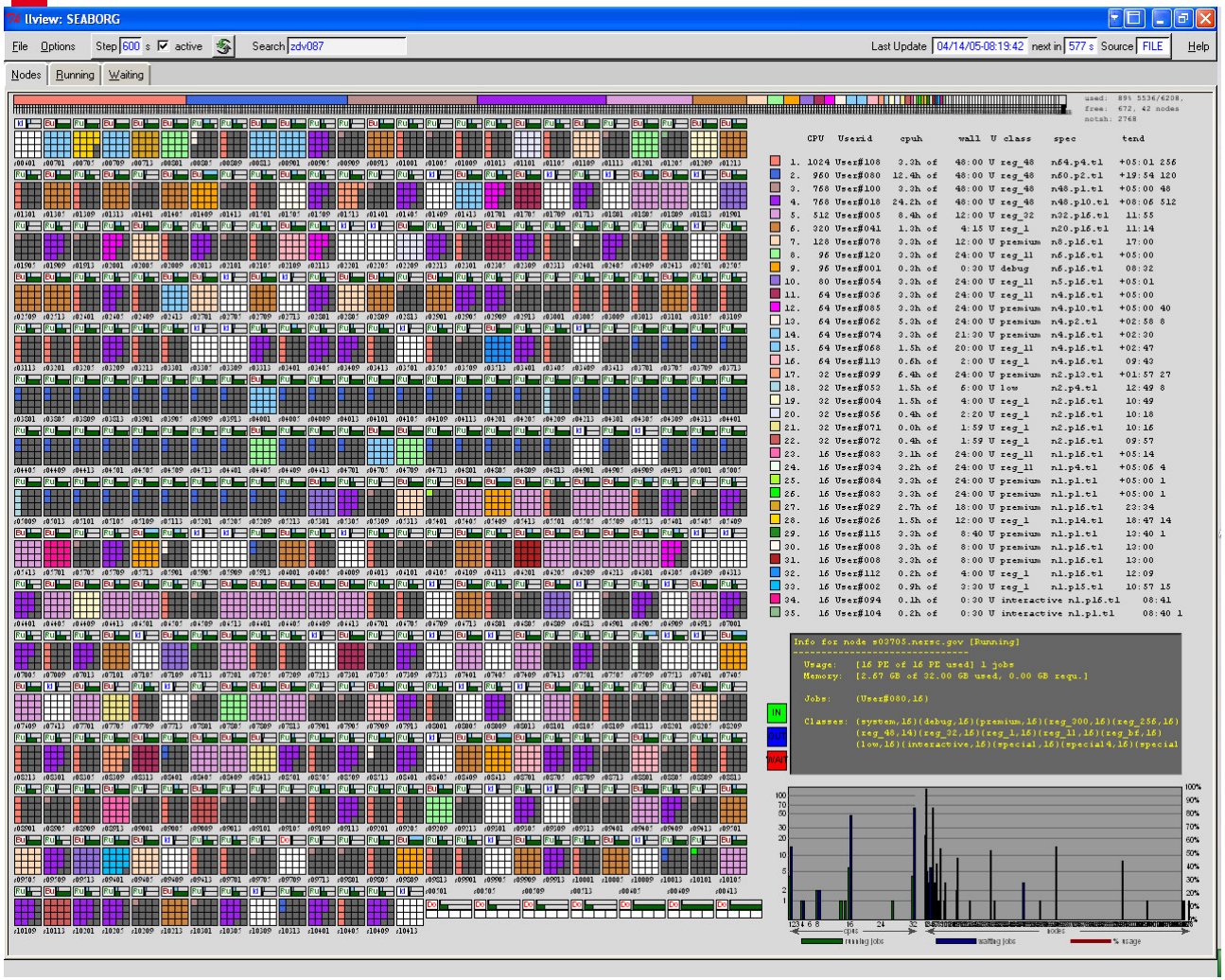
IBM p690 Power4 system, 41 nodes, 32 CPUs/node, LoadLeveler

Example 2: Jubl



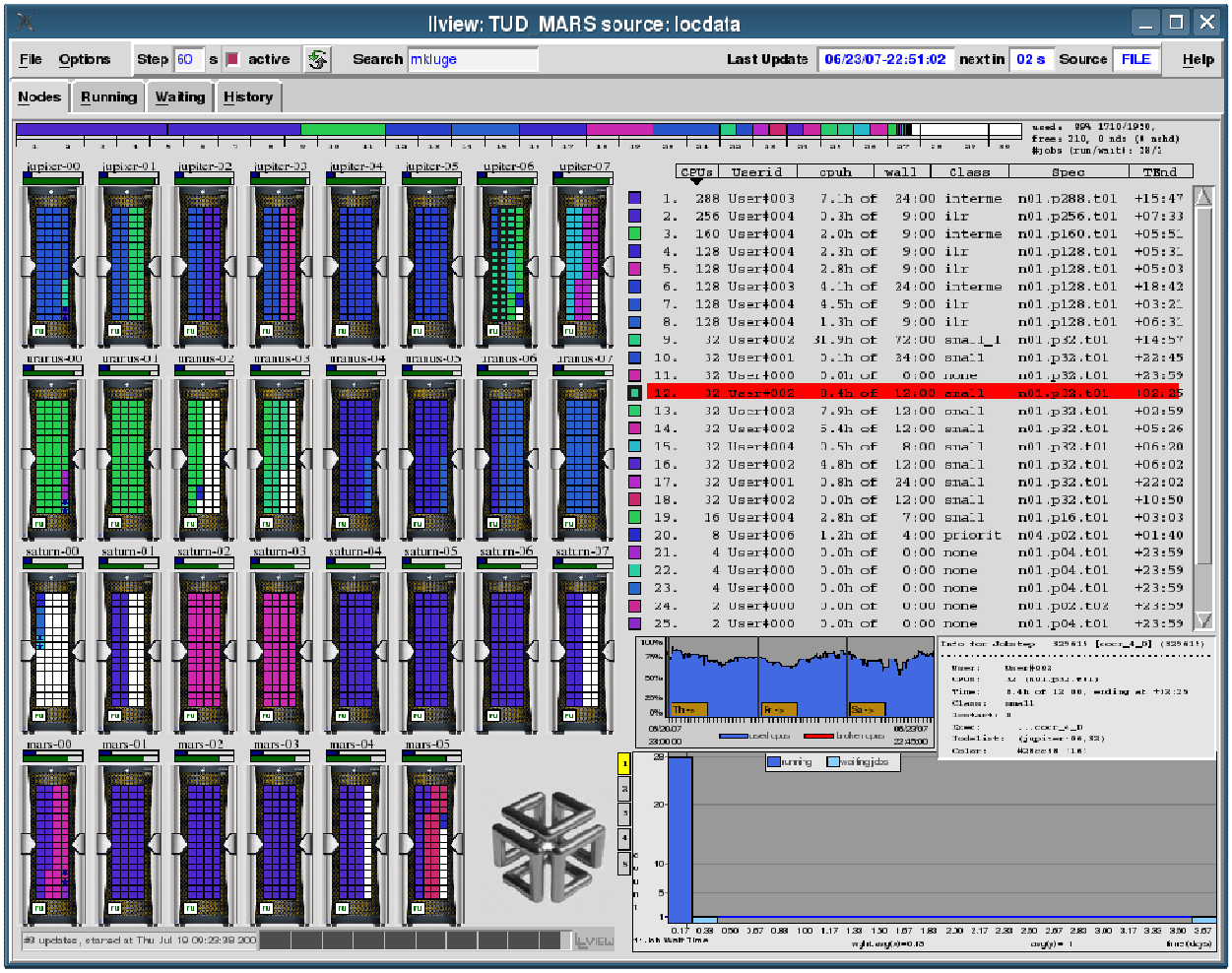
IBM BlueGene/L,
8 racks 16384 CPUs,
LoadLeveler+DB2

Example 3: Seaborg



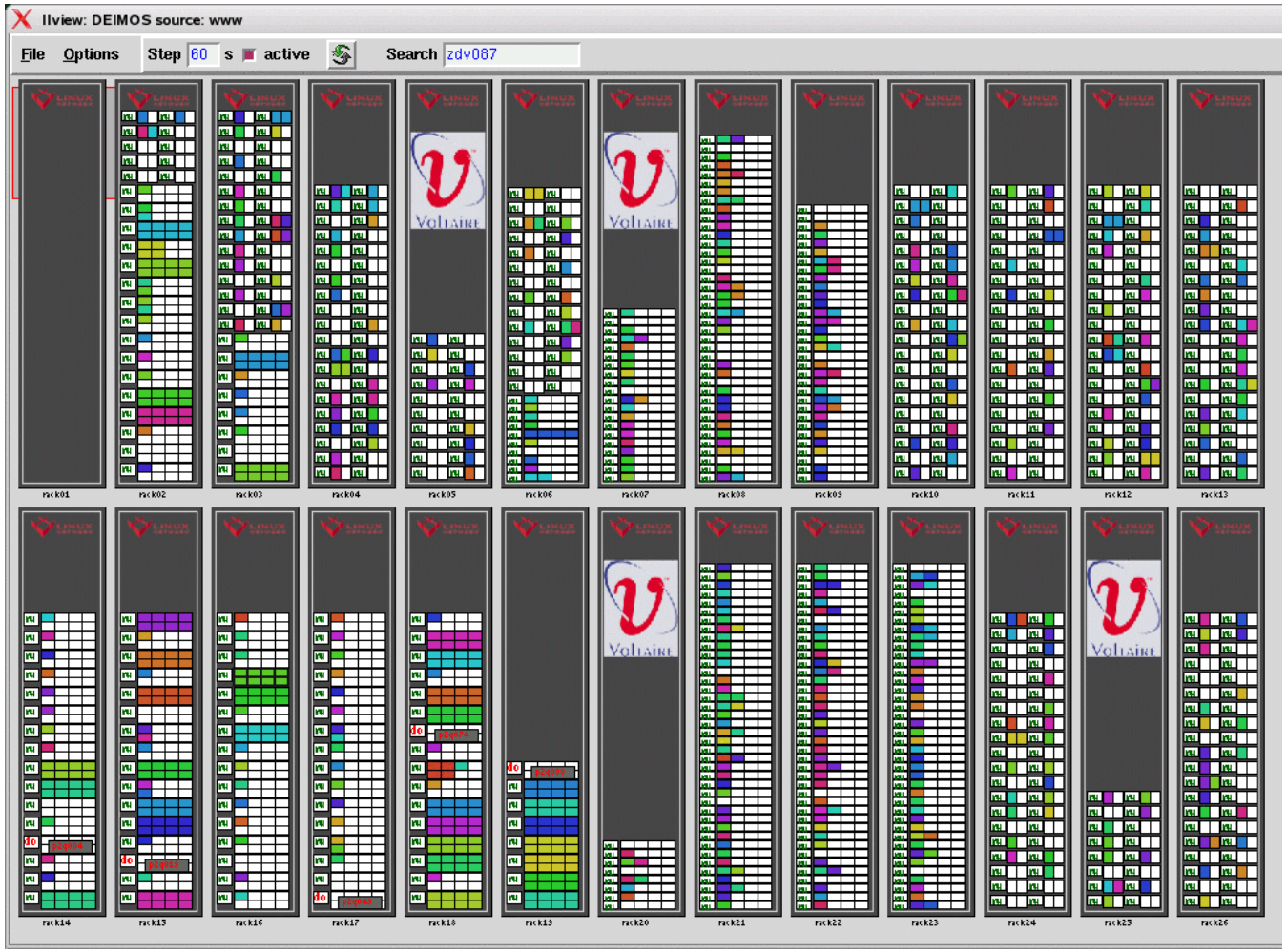
Seaborg, NERSC
IBM Power 3
389 nodes (16 PE)
6656 CPUs

Example 4: Altix



TU Dresden
SGI Altix 4700
2048 cores
32 (30) racks

Example 4: PC-farm



TU Dresden,
Deimos PC-farm,
726 nodes, 2584 CPUs,
#jobs >> 1000